EXHIBIT 18

lightninglabs / taro Public

A layer 1 daemon, for the Taro protocol specification, written in Go (golang)

MIT license

233 stars 45 forks

Star Notifications

Code Issues 61 Pull requests 10 Discussions Actions Projects 2 Security Ins

main Go to file

Roasbeef Merge pull request #268 from positiveblue/new-server-signature ... yesterday 801

View code

Taro

The Taro Daemon `tarod` implements the Taro protocol for issuing assets on the Bitcoin blockchain. Taro leverages Taproot transactions to commit to newly created assets and their transfers in an efficient and scalable manner. Multiple assets can be created and transferred in a single bitcoin UTXO, while witness data is transacted and kept off-chain.

Features:

- Mint assets
- Send and receive assets
- Export and import Taro proofs
- Create and manage profiles

How it works:

When minting a new asset, Taro will generate the relevant witness data, assign the asset to a key held by you and publish the corresponding bitcoin UTXO -- the minting transaction.

The outpoint this minting transaction consumes becomes the `genesis_point` of the newly minted asset, acting as its unique identifier. Assets can be spent to a new recipient, who provides the sender with the necessary information encoded in their Taro address.

To transact assets, the witnesses in the prior Taro transaction are recommitted into one or multiple taproot outputs while the necessary witness data is passed to the recipient. Similar to bitcoin transactions, the remaining balance is spent back to the sender as a change output.

Learn more about the Taro protocol.

Architecture:

Taro is implemented as the Taro Daemon `tarod` and the Taro Command Line Interface `tarocli`. Additionally, `tarod` exposes a GRPC interface to allow for a direct integration into applications.

Taro leverages several LND features including the Taproot wallet and signing capabilities. These facilities are accessed through LND's GRPC.

The Taro stack:

```
Bitcoin blockchain backend <-> LND <-> Taro
```

Custody of Taro assets is segmented across LND and Taro to maximize security. LND holds the private key, which has had a taproot tweak applied to it, controlling the bitcoin UTXO holding the Taro asset. The taproot tweak on the other hand is held by Taro. This increases the requirements for asset recovery as both the internal key as well as the taproot tweak are necessary to spend the output. This prevents LND from accidentally burning Taro assets.

Prerequisites:

Taro requires LND (compiled on the latest `master` branch with the relevant tags, see below) to be synced and running on the same Bitcoin network as Taro (e.g. regtest, simnet, testnet3). RPC connections need to be accepted and a valid macaroon needs to be present.

```
git clone https://github.com/lightningnetwork/lnd.git
cd lnd
make install tags="signrpc walletrpc chainrpc invoicesrpc"
```

Installation:

From source:

Compile Taro from source by cloning this repository. Go version 1.18 or higher is required.

README.md

```
cd taro
make install
```

Initialization:

Run Taro with the command `tarod`. Specify how Taro can reach LND and what network to run Taro with by passing it additional flags.

```
# Ensure lnd and its bitcoind/btcd backend are running first.
tarod --network=testnet --debuglevel=debug --lnd.host=localhost:10009 --lnd.macaroonpath=~/.lnd/data/
```

Usage:

See a full list of options by executing:

```
tarod --help
```

Use `tarocli` to interact with `tarod`

```
tarocli assets mint --type normal --name fantasycoin --supply 100 --meta "fantastic money" --skip_bat
```

```
tarocli assets list
```

```
tarocli addrs new --genesis_bootstrap_info bab08407[...]129bf6d0 --amt 21
```

```
tarocli assets send --addr tarotb1q[...]tywpre3a
```

Development

API

Taro exposes a GRPC (port 10029) and a REST (port 8089) API. Connections are encrypted with TLS and authenticated using macaroons. The API is documented here. Further guides can be found here.

Mainnet

The current codebase does not support the Bitcoin `mainnet`. Patching the code to run on `mainnet` will very likely lead to loss of funds (both the minted assets and the BTC UTXO) as things will break or change in the future.

Submit feature requests

The GitHub issue tracker can be used to request specific improvements or report bugs.

Join us on Slack

Join us in the Lightning Labs Slack and join the `#taro` channel to ask questions and interact with the community.

Releases

2 tags

Packages

No packages published

Used by 4

@habibitcoin / **habibtaro**

@AreaLayer / **Tarot**

@jacohend / **ltd**

Contributors 15

+ 4 contributors

Languages

Go 98.5% Other 1.5%